# Robotic disassembly sequence planning using enhanced discrete bees algorithm in remanufacturing

Liu, Jiayi; Zhou, Zude; Pham, Duc Truong; Xu, Wenjun; Ji, Chunqian; Liu, Quan

*DOI:*
10.1080/00207543.2017.1412527



*Document Version*
Peer reviewed version

*Citation for published version (Harvard):*
Liu, J, Zhou, Z, Pham, DT, Xu, W, Ji, C & Liu, Q 2018, 'Robotic disassembly sequence planning using enhanced discrete bees algorithm in remanufacturing', *International Journal of Production Research*, vol. 56, no. 9, pp. 3134-3151. https://doi.org/10.1080/00207543.2017.1412527

# Robotic Disassembly Sequence Planning using Enhanced Discrete Bees Algorithm in Remanufacturing

Jiayi Liu[1,2], Zude Zhou[1,2], Duc Truong Pham[3], Wenjun Xu[1,4,*],

Chunqian Ji[3], Quan Liu[1,2]

[1]*School of Information Engineering, Wuhan University of Technology, Wuhan 430070, China*

[2]*Key Laboratory of Fiber Optic Sensing Technology and Information Processing (Wuhan University of Technology), Ministry of Education, Wuhan 430070, China*

[3]*Department of Mechanical Engineering, University of Birmingham, Birmingham B15 2TT, UK*

[4]*Hubei Key Laboratory of Broadband Wireless Communication and Sensor Networks (Wuhan University of Technology), Wuhan 430070, China*

Email: jyliu@whut.edu.cn; zudezhou@whut.edu.cn; d.t.pham@bham.ac.uk; xuwenjun@whut.edu.cn; c.ji@bham.ac.uk; quanliu@whut.edu.cn

# Robotic Disassembly Sequence Planning using Enhanced Discrete Bees Algorithm in Remanufacturing

Increasing attention is being paid to remanufacturing due to environmental protection and resource saving. Disassembly, as an essential step of remanufacturing, is always manually finished which is time-consuming while robotic disassembly can improve disassembly efficiency. Before the execution of disassembly, generating optimal disassembly sequence plays a vital role in improving disassembly efficiency. In this paper, to minimize the total disassembly time, an enhanced discrete Bees algorithm (EDBA) is proposed to solve robotic disassembly sequence planning (RDSP) problem. Firstly, the modified feasible solution generation (MFSG) method is used to build the disassembly model. After that, the evaluation criterions for RDSP are proposed to describe the total disassembly time of a disassembly sequence. Then, with the help of mutation operator, EDBA is proposed to determine the optimal disassembly sequence of RDSP. Finally, case studies based on two gear pumps are used to verify the effectiveness of the proposed method. The performance of EDBA is analyzed under different parameters and compared with existing optimization algorithms used in disassembly sequence planning (DSP). The result shows the proposed method is more suitable for robotic disassembly than the traditional method and EDBA generates better quality of solutions compared with the other optimization algorithms.

Keywords: remanufacturing; robotic disassembly sequence planning; enhanced discrete bees algorithm; disassembly sequence planning; intelligent optimization

## 1. Introduction

The traditional manufacturing industry has the disadvantages of low resource utilization and high environmental pollution. Cloud manufacturing (Tao et al. 2017a, 2017b) and

remanufacturing (Diallo et al. 2017) *etc.* are regarded as the future trends of manufacturing industry, they can make full use of manufacturing resources (Tao et al. 2008) such as manufacturing equipments, manufactured products *etc.* Improper handling of the End-of-Life (EoL) products which have been used for many years usually leads to environmental pollution and resources-wasting (Ren et al. 2017). Remanufacturing takes both environmental protection and economic development into considerations by reusing EoL products (Guide 2000). When EoL products need to be remanufactured, disassembly should be firstly considered. Due to the complexities of disassembly process, disassembly process is always manually finished. Recently, robotic disassembly has been paid much attention due to high efficiency. The cognitive robot was proposed to handle uncertainties in dynamic disassembly process (Vongbunyong, Kara, and Pagnucco 2012). Afterwards, the basic behavior control (Vongbunyong, Kara, and Pagnucco 2013a) and the advanced behavior control strategies (Vongbunyong, Kara, and Pagnucco 2015) of cognitive robot were proposed. Based on LCD screens, robotic disassembly system which consists of reasoning, execution monitoring, learning/revision strategy (Vongbunyong, Kara, and Pagnucco 2013b) was proposed to realize automated disassembly.

The disassembly process mainly contains two parts: disassembly planning and disassembly execution. In the disassembly planning, obtaining the optimal disassembly sequence plays a vital role in reducing disassembly time and disassembly cost, *etc.* (Luo, Peng, and Gu 2016). Many researchers have studied DSP problem to find the optimal disassembly sequence. However, most of existing researches focus on solving DSP for

manual disassembly. For RDSP, the traditional DSP model is not adaptable because of different characteristics of humans and robots. To avoid the obstacle caused by contour of EoL products, the moving path of industrial robot's end-effector should be considered. It also has influence on the total disassembly time. In this paper, an optimization algorithm named EDBA is proposed to solve RDSP to minimize the total disassembly time.

The rest of this paper is organized as follows: firstly, we briefly review the related works in Section 2. After that, feasible disassembly sequence is obtained by modified space interference matrix. In Section 4, in order to minimize the total disassembly time, the evaluation criterions for RDSP are proposed. EDBA is proposed to solve RDSP in Section 5. Furthermore, case studies based on two gear pumps are used to verify the proposed method. The performance of EDBA is analyzed under different parameters and compared with some existing optimization algorithms. Finally, conclusions are made in Section 7.

## 2. Related works

Nowadays, there are many researches focus on DSP problems. In the reference (Xing, Wang, and Liu 2012), the changes of disassembly direction, total disassembly distance and length of disassembly sequence were simultaneously considered. DSP was solved by ant colony algorithm. Considering the changes of disassembly tool, the changes of disassembly direction, the part volume and the maintainability, Kheder used genetic algorithm to obtain optimal disassembly sequence of a rear axle (Kheder, Trigui, and Aifaoui 2015). To disassemble heavy, hazardous and high-value components as early as

possible, brute-force method was proposed to obtain the optimal disassembly sequence based on waste electrical and electronic equipments (Jin et al. 2015). To simultaneously optimize the disassembly level, recovery options and disassembly sequence, an improved co-evolutionary algorithm was used to find the optimal disassembly solutions (Meng et al. 2016). In the parallel disassembly environment, an integer programming model and an optimal branch and bound algorithm were used to minimize the operation cost and the sequence-dependent set-ups under selective disassembly mode (Kim and Lee 2017).

When DSP is considered together with robotic disassembly, the characteristics of industrial robots should be considered. The total disassembly time, as the major optimization objective in RDSP, contains four parts: basic disassembly time (Song et al. 2014), penalty time of disassembly direction change, penalty time of disassembly tool change (Xia et al. 2014a) and moving time of the end-effector between different disassembly points (ElSayed et al. 2011). To disassemble personal computers, ElSayed used genetic algorithm to get optimal disassembly sequence for robotic disassembly (ElSayed, Kongar, and Gupta 2010). After that, an online genetic algorithm was used to solve DSP, it can handle dynamical disassembly process (ElSayed et al. 2012). These researches consider range-sensing camera, image segmentation algorithm and movement of industrial robot’s end-effector in the disassembly process. For the movement of industrial robot’s end-effector, the moving time between different disassembly points (ElSayed et al. 2011) was a part of total disassembly time, it was calculated by Euclidean distance between different disassembly points and moving

speed of the end-effector. After that, Alshibili et al. (2015) used tabu search to obtain optimal disassembly sequence, the moving time between different disassembly points was also calculated by the same way. However, during robotic disassembly process, the moving path between different disassembly points should not be straight-line path, the obstacle-avoidance moving path of the end-effector should be considered to avoid physical collisions.

To obtain optimal disassembly sequence, optimization algorithms such as Genetic Algorithm (GA) (Lambert 2003), Ant Colony Optimization (ACO) *etc.* are usually used. In reference (Go et al. 2012), the disassembly sequences were coded into chromosomes and the selection/mutation/crossover operators were used. Based on a machine vise, Xu used adaptive particle swarm optimization to solve DSP, the inertia weight and mutation probabilities were adaptively determined (Xu, Zhang, and Fei 2011). With the help of multi-layer representation method, based on a single-speed reduction gearbox, the optimal disassembly sequence was obtained by ACO (Luo, Peng, and Gu 2016). Bees algorithm (BA) (Pham and Ghanbarzadeh 2007) is an optimization algorithm inspired by the foraging behavior of bees. Compared with existing optimization algorithms, BA has its great competitiveness (Yuce et al. 2013). To the best of our knowledge, BA has not been used in RDSP field yet.

## 3. Disassembly model

After years of usage, the disassembly precedence relationship of EoL product may be different from its original status. For the EoL product with unknown structure, it is difficult to build its disassembly model in advance. In this paper, the following

assumptions are made: 1. the proposed method is applicable to repetitive disassembly of the same product with known components and geometric information (Xia et al. 2014a); 2. All parts of EoL product can be disassembled through corresponding disassembly operations.

To obtain feasible disassembly sequence, disassembly model should be firstly established to describe the disassembly precedence relationships. In the existing researches, the disassembly model is mainly established by graph-based methods (Tian, Zhou, and Chu 2013), Petri net methods (Xia et al. 2014b) and matrix-based methods (Percoco and Diella 2013) *etc.* In robotic disassembly, the disassembly direction of each part should be provided for the industrial robot. In this paper, MFSG which contains the modified space interference matrix and the interference matrix analysis is used to build disassembly model.

### *3.1 Modified space interference matrix*

Jin, Li, and Xia (2013) used space interference matrices along six directions (X+, X-, Y+, Y-, Z+, Z-) to describe disassembly precedence relationships between different parts. In the space interference matrix $S_{md}$, element $s_{ij}$ indicates whether component $j$ impedes the movement of component $i$ along $md$ ($md$ = X+, X-, Y+, Y-, Y+ or Y-) direction. If component $j$ impedes the movement of component $i$ along $md$ direction, $s_{ij}$ is 1, otherwise $s_{ij}$ is 0. In their works, space interference matrix along negative axis (such as X-) was regarded as the transposed matrix of space interference matrix along corresponding positive axis (such as X+). However, when the bolt is considered, the space interference matrix needs to be modified. A simple case is studied as shown in

Figure 1.

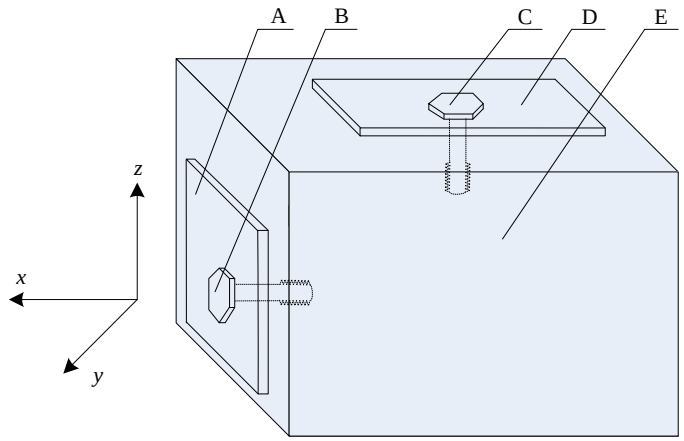



Figure 1. A simple case for modified space interference matrix

$$S_{x+}=\begin{array}{c}\\A\\B\\C\\D\\E\end{array}\begin{array}{c}\begin{array}{ccccc}A&B&C&D&E\end{array}\\\begin{bmatrix}0&1&0&0&0\\0&0&0&0&0\\0&0&0&1&1\\0&0&1&0&0\\1&1&1&0&0\end{bmatrix}\end{array};\quad S_{y+}=\begin{array}{c}\\A\\B\\C\\D\\E\end{array}\begin{array}{c}\begin{array}{ccccc}A&B&C&D&E\end{array}\\\begin{bmatrix}0&1&0&0&0\\1&0&0&0&1\\0&0&0&1&1\\0&0&1&0&0\\0&1&1&0&0\end{bmatrix}\end{array};\quad S_{z+}=\begin{array}{c}\\A\\B\\C\\D\\E\end{array}\begin{array}{c}\begin{array}{ccccc}A&B&C&D&E\end{array}\\\begin{bmatrix}0&1&0&0&0\\1&0&0&0&1\\0&0&0&0&0\\0&0&1&0&0\\0&1&1&1&0\end{bmatrix}\end{array}$$

$$S_{x-}=\begin{array}{c}\\A\\B\\C\\D\\E\end{array}\begin{array}{c}\begin{array}{ccccc}A&B&C&D&E\end{array}\\\begin{bmatrix}0&0&0&0&1\\1&0&0&0&1\\0&0&0&1&1\\0&0&1&0&0\\0&0&1&0&0\end{bmatrix}\end{array};\quad S_{y-}=\begin{array}{c}\\A\\B\\C\\D\\E\end{array}\begin{array}{c}\begin{array}{ccccc}A&B&C&D&E\end{array}\\\begin{bmatrix}0&1&0&0&0\\1&0&0&0&1\\0&0&0&1&1\\0&0&1&0&0\\0&1&1&0&0\end{bmatrix}\end{array};\quad S_{z-}=\begin{array}{c}\\A\\B\\C\\D\\E\end{array}\begin{array}{c}\begin{array}{ccccc}A&B&C&D&E\end{array}\\\begin{bmatrix}0&1&0&0&0\\1&0&0&0&1\\0&0&0&1&1\\0&0&0&0&1\\0&1&0&0&0\end{bmatrix}\end{array} \tag{1}$$

$$S_{x,y,z}=\begin{array}{c}\\A\\B\\E\end{array}\begin{array}{c}\begin{array}{ccc}A&B&E\end{array}\\\begin{bmatrix}000000&101111&010000\\011111&000000&011111\\100000&101111&000000\end{bmatrix}\end{array}\quad\begin{array}{c}result\\111111\\011111\\101111\end{array} \tag{2}$$

$$S_{x+}=\begin{array}{c}\\A\\B\\C\\D\\E\end{array}\begin{array}{c}\begin{array}{ccccc}A&B&C&D&E\end{array}\\\begin{bmatrix}0&1&0&0&0\\0&0&0&0&0\\0&0&0&1&1\\0&0&1&0&0\\1&1&1&0&0\end{bmatrix}\end{array};\quad S_{y+}=\begin{array}{c}\\A\\B\\C\\D\\E\end{array}\begin{array}{c}\begin{array}{ccccc}A&B&C&D&E\end{array}\\\begin{bmatrix}0&1&0&0&0\\1&0&0&0&1\\0&0&0&1&1\\0&0&1&0&0\\0&1&1&0&0\end{bmatrix}\end{array};\quad S_{z+}=\begin{array}{c}\\A\\B\\C\\D\\E\end{array}\begin{array}{c}\begin{array}{ccccc}A&B&C&D&E\end{array}\\\begin{bmatrix}0&1&0&0&0\\1&0&0&0&1\\0&0&0&0&0\\0&0&1&0&0\\0&1&1&1&0\end{bmatrix}\end{array}$$

$$S_{x-}=\begin{array}{c}\\A\\B\\C\\D\\E\end{array}\begin{array}{c}\begin{array}{ccccc}A&B&C&D&E\end{array}\\\begin{bmatrix}0&1&0&0&1\\1&0&0&0&1\\0&0&0&1&1\\0&0&1&0&0\\0&1&1&0&0\end{bmatrix}\end{array};\quad S_{y-}=\begin{array}{c}\\A\\B\\C\\D\\E\end{array}\begin{array}{c}\begin{array}{ccccc}A&B&C&D&E\end{array}\\\begin{bmatrix}0&1&0&0&0\\1&0&0&0&1\\0&0&0&1&1\\0&0&1&0&0\\0&1&1&0&0\end{bmatrix}\end{array};\quad S_{z-}=\begin{array}{c}\\A\\B\\C\\D\\E\end{array}\begin{array}{c}\begin{array}{ccccc}A&B&C&D&E\end{array}\\\begin{bmatrix}0&1&0&0&0\\1&0&0&0&1\\0&0&0&1&1\\0&0&1&0&1\\0&1&1&0&0\end{bmatrix}\end{array} \tag{3}$$

For this case, the space interference matrices along six directions are described by Equation (1). According to interference matrix analysis mentioned in Section 3.2, if both bolt C and component D have been removed along Z+ direction, the traditional space interference matrix $S_{x,y,z}$ is shown in Equation (2). For the remaining parts ABE, based on interference matrix analysis and Equation (2), component E can be removed along X- direction before bolt B is removed. However, it is at variance with the reality. Because for the practical disassembly process, component E can be removed only after bolt B has been disassembled along X+ direction. Thus, each interference matrix should be considered separately instead of using the transposed matrix. In the modified interference matrix $S_{md}$, element $s_{ij}$ indicates whether component *i* can be disassembled along *md* direction when component *j* exists, if component *i* can be disassembled along *md* direction when component *j* exists, element $s_{ij}$ is 0, otherwise, it is 1. The modified interference matrices are expressed by Equation (3). For example, element $S_{BE}$ is 0 in the modified space interference matrix $S_{x+}$, although component E has contact relationships with bolt B, bolt B can be removed along X+ direction by unscrewing operations. Element $S_{EB}$ is 1 in the interference matrix $S_{x-}$, because component E can not be removed along X- direction before bolt B has been removed.

### *3.2 Interference matrix analysis*

Based on the modified space interference matrix, interference matrix analysis is used to obtain the feasible disassembly sequences. Interference matrices $S_{x+}$, $S_{x-}$, $S_{y+}$, $S_{y-}$, $S_{z+}$, $S_{z-}$ are integrated into interference matrix $S_{x,y,z}$ as shown in Equation (4) by the following method. The element $s_{x,y,z(i,j)}$ in the integrated matrix $S_{x,y,z}$ is a string of six digits of 0

and 1 representing the elements $s_{ij}$ of the six interference matrices listed in the order of entries in matrices $S_{x+}$, $S_{x-}$, $S_{y+}$, $S_{y-}$, $S_{z+}$ and $S_{z-}$ respectively. The Boolean operator 'OR' acts on each row of interference matrices ($S_{x+}$, $S_{x-}$, $S_{y+}$, $S_{y-}$, $S_{z+}$, $S_{z-}$) to obtain the column *result* as shown in Equation (4). For instance, the third element of column *result* is 111101, each bit is calculated by Boolean operator acts on the third row of each interference matrix (in order of $S_{x+}$, $S_{x-}$, $S_{y+}$, $S_{y-}$, $S_{z+}$, $S_{z-}$). Each element of column *result* has 6 bits, if the first bit is 0, it means this component can be disassembled along X+ direction, otherwise, it can not (first bit for X+ direction, second bit for X- direction, third bit for Y+ direction, *etc.*). In the array '111101', the fifth bit is 0, it means bolt C can be disassembled along Z+ direction. Thus, according to this rule and Equation (4), bolts B and C can be disassembled along X+ direction and Z+ direction respectively. If bolt B has been disassembled along X+ direction, the second column and the second row of $S_{x,y,z}$ in Equation (4) are deleted as shown in Equations (5).

$$S_{x,y,z} = \begin{array}{c} \\ A \\ B \\ C \\ D \\ E \end{array} \begin{array}{c} \begin{array}{ccccc} A & B & C & D & E \end{array} \\ \begin{bmatrix} 000000 & 111111 & 000000 & 000000 & 010000 \\ 011111 & 000000 & 000000 & 000000 & 011111 \\ 000000 & 000000 & 000000 & 111101 & 111101 \\ 000000 & 000000 & 111111 & 000000 & 000001 \\ 100000 & 111111 & 111111 & 000010 & 000000 \end{bmatrix} \end{array} \begin{array}{c} result \\ 111111 \\ 011111 \\ 111101 \\ 111111 \\ 111111 \end{array} \tag{4}$$

$$S_{x,y,z} = \begin{array}{c} \\ A \\ C \\ D \\ E \end{array} \begin{array}{c} \begin{array}{cccc} A & C & D & E \end{array} \\ \begin{bmatrix} 000000 & 000000 & 000000 & 010000 \\ 000000 & 000000 & 111101 & 111101 \\ 000000 & 111111 & 000000 & 000001 \\ 100000 & 111111 & 000010 & 000000 \end{bmatrix} \end{array} \begin{array}{c} result \\ 010000 \\ 111101 \\ 111111 \\ 111111 \end{array} \tag{5}$$

$$S_{x,y,z} = \begin{array}{c} \\ A \\ D \\ E \end{array} \begin{array}{c} \begin{array}{ccc} A & D & E \end{array} \\ \begin{bmatrix} 000000 & 000000 & 010000 \\ 000000 & 000000 & 000001 \\ 100000 & 000010 & 000000 \end{bmatrix} \end{array} \begin{array}{c} result \\ 010000 \\ 000001 \\ 100010 \end{array} \tag{6}$$

$$S_{x,y,z} = \begin{array}{c} \\ D \\ E \end{array} \begin{array}{c} \begin{array}{cc} D & E \end{array} \\ \begin{bmatrix} 000000 & 000001 \\ 000010 & 000000 \end{bmatrix} \end{array} \begin{array}{c} result \\ 000001 \\ 000010 \end{array} \tag{7}$$

From Equation (5), it is obvious that component A can be disassembled along X+, Y+, Y-, Z+ or Z- direction, component C can be disassembled along Z+ direction. If component C has been removed along Z+ direction, the corresponding column and row of $S_{x,y,z}$ in Equation (5) are deleted. Then the interference matrix $S_{x,y,z}$ is described by Equation (6). It is obvious component A can be removed along X+, Y+, Y-, Z+ or Z- direction, component D can be removed along X+, X-, Y+, Y- or Z+ direction, component E can be removed along X-, Y+, Y-, or Z- direction. If component A has been removed along X+ direction, the interference matrix $S_{x,y,z}$ is obtained as shown in Equation (7). From Equation (7), component D can be removed along X+, X-, Y+, Y- or Z+ direction and component E can be removed along X+, X-, Y+, Y- or Z- direction. If component D has been removed along Z+ direction, the remaining component E can be removed along any direction (We choose Z+ direction here). The feasible disassembly sequence is B/C/A/D/E, the corresponding disassembly direction is X+/Z+/X+/Z+/Z+. During the generation of disassembly solutions (Equations (4) through (7)), the disassembly solutions are not unique. Many other alternative disassembly solutions can also be generated by the same method. Without capturing these alternatives, high quality of disassembly solutions could be missed.

## 4. Evaluation criterions for RDSP

In this paper, the optimization objective of RDSP is to minimize the total disassembly time for disassembling an EoL product. The total disassembly time mainly contains the basic disassembly time, the penalty time for disassembly direction changes, the penalty time for disassembly tool changes and the moving time between different disassembly

points.

For the basic disassembly time, it is described as disassembling a component by the industrial robots (separating a component, unscrewing a screw *etc.*). In this paper, the basic disassembly time for disassembling each component is assumed to be constant (Luo, Peng, and Gu 2016).

During the disassembly process, to deal with disassembly direction changes, it takes additional time for the industrial robot to adjust its posture. In this paper, the penalty time for disassembly direction change is directly added to the total disassembly time, it is expressed by Equation (8).

$$dt(x_i, x_{i+1}) = \begin{cases} 0 & \text{direction is not changed} \\ 1 & \text{direction is changed by } 90^\circ \\ 2 & \text{direction is changed by } 180^\circ \end{cases} \tag{8}$$

$$TT = \begin{array}{c} \\ Sp \\ Sc \\ Gr \\ Pl \\ EC \\ Ha \end{array} \begin{array}{c} \begin{array}{cccccc} Sp & Sc & Gr & Pl & EC & Ha \end{array} \\ \begin{bmatrix} TT_1 & tt_{1,2} & tt_{1,3} & tt_{1,4} & tt_{1,5} & tt_{1,6} \\ tt_{2,1} & TT_2 & tt_{2,3} & tt_{2,4} & tt_{2,5} & tt_{2,6} \\ tt_{3,1} & tt_{3,2} & TT_3 & tt_{3,4} & tt_{3,5} & tt_{3,6} \\ tt_{4,1} & tt_{4,2} & tt_{4,3} & TT_4 & tt_{4,5} & tt_{4,6} \\ tt_{5,1} & tt_{5,2} & tt_{5,3} & tt_{5,4} & TT_5 & tt_{5,6} \\ tt_{6,1} & tt_{6,2} & tt_{6,3} & tt_{6,4} & tt_{6,5} & TT_6 \end{bmatrix} \end{array} \tag{9}$$

$$TT_1 = \begin{array}{c} \\ M1 \\ M2 \\ M3 \\ \dots \\ Mn \end{array} \begin{array}{c} \begin{array}{ccccc} M1 & M2 & M3 & \dots & Mn \end{array} \\ \begin{bmatrix} 0 & tta_{1,2} & tta_{1,3} & \dots & tta_{1,n} \\ tta_{2,1} & 0 & tta_{2,3} & \dots & tta_{2,n} \\ tta_{3,1} & tta_{3,2} & 0 & \dots & tta_{3,n} \\ \dots & \dots & \dots & \dots & \dots \\ tta_{n,1} & tta_{n,2} & tta_{n,3} & \dots & 0 \end{bmatrix} \end{array} \tag{10}$$

Disassembly tool changes also cause more disassembly time for the industrial robots. From Figure 2(a), different disassembly tools are used for different disassembly operations. The penalty time for disassembly tool change is described by Equation (9) (*Sp, Sc, Gr, Pl, EC, Ha* mean spanner, screwdriver, gripper, plier, electrical cutting and hammer respectively). For the same disassembly operation, to disassemble different

components, different disassembly tools need to be considered. For example, when unscrewing operation is considered, to disassemble different bolts (M1, M2, M3 *etc.*), different spanners (M1, M2, M3 *etc.*) need to be considered, the corresponding penalty time of disassembly tool change is described by Equation (10).

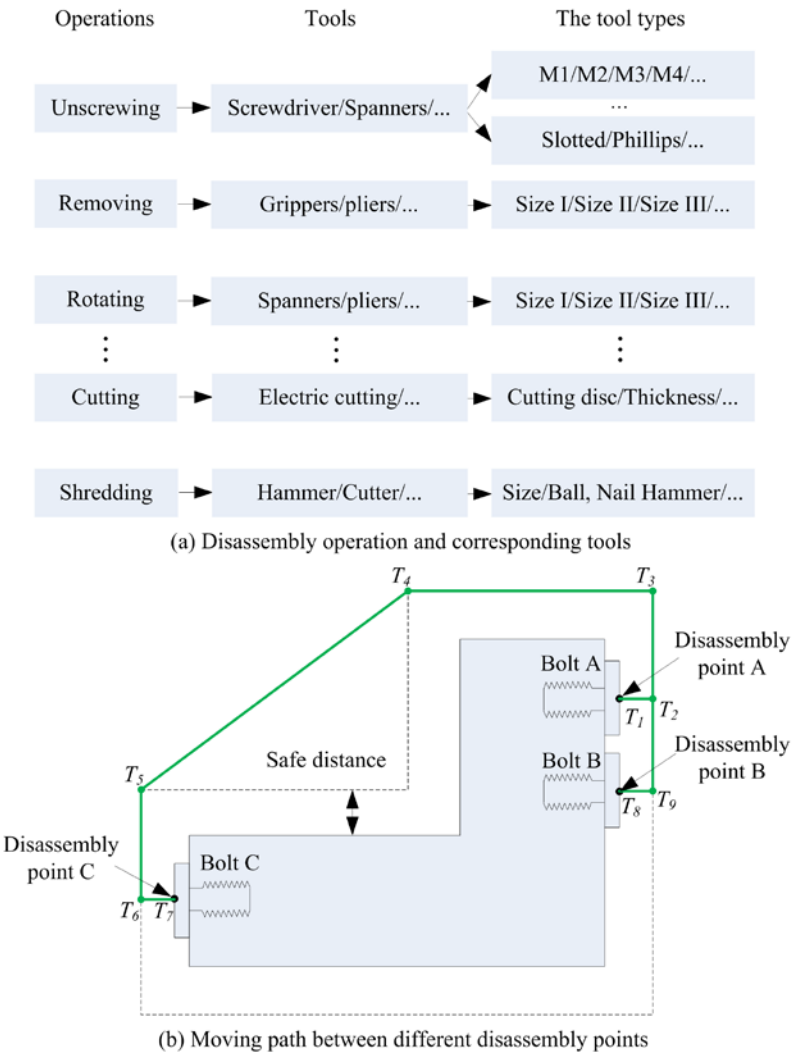



Figure 2. The disassembly tools and moving path between different disassembly points

Strictly, to calculate the moving time, the obstacle-avoidance path should be firstly considered, after that, the collision free trajectory planning should also be considered (Constantinescu and Croft 2000). The moving time of industrial robot's end-effector between different disassembly points is decided by not only the complexity of EoL

products, position of the product in the robot workspace, but also the types of industrial robots. All the factors have impacts on the moving time between different disassembly points and should be considered in robotic disassembly. In this paper, for simplicity, the moving time is calculated by the length of moving path between different disassembly points and the linear velocity of industrial robot's end-effector. Assumption 1 mentioned in Section 3 ensures the geometric and structure information of EoL product can be provided in advance. Thus, the length of moving path between different disassembly points can be decided in advance so that industrial robot' end-effector can move along the predefined path. Besides, assumption 2 ensures no non-removable part exists. A simple case is considered as shown in Figure 2(b). Firstly, the safe distance needs to be considered to ensure that the industrial robot's end-effector can move without collisions with the product. The contour of safe moving path is described by the dotted line in Figure 2(b). In addition, the length matrix *MP* is described by Equation (11), the length of moving path between point A and point B, point A and point C, point B and point C are expressed as $a_{12}$, $a_{13}$, $a_{23}$ respectively.

$$MP = \begin{array}{c} \\ A \\ B \\ C \end{array} \begin{array}{c} \begin{array}{ccc} A & B & C \end{array} \\ \begin{bmatrix} 0 & a_{12} & a_{13} \\ a_{12} & 0 & a_{23} \\ a_{13} & a_{23} & 0 \end{bmatrix} \end{array} \tag{11}$$

$$a_{12} = T_1T_2 + T_2T_9 + T_9T_8$$

$$a_{13} = T_1T_2 + T_2T_3 + T_3T_4 + T_4T_5 + T_5T_6 + T_6T_7$$

$$a_{23} = T_8T_9 + T_9T_2 + T_2T_3 + T_3T_4 + T_4T_5 + T_5T_6 + T_6T_7$$

After the length matrix *MP* is determined, the moving time between different disassembly points is calculated by Equation (12) ($v_{end\text{-}effector}$ is the line velocity of

industrial robot's end-effector).

$$mt(x_i, x_{i+1}) = MP(x_i, x_{i+1}) / v_{end-effector} \tag{12}$$

In summary, the total disassembly time to disassemble an EoL product is calculated by Equation (13).

$$f(X) = \sum_{i=0}^{n-1} bt(x_i) + \sum_{i=0}^{n-2} dt(x_i, x_{i+1}) + \sum_{i=0}^{n-2} tt(x_i, x_{i+1}) + \sum_{i=0}^{n-2} mt(x_i, x_{i+1}) \tag{13}$$

Where $n$ represents the number of total parts, $bt(x_i)$ means the basic disassembly time for disassembling part $x_i$, while $dt(x_i, x_{i+1})$, $tt(x_i, x_{i+1})$ and $mt(x_i, x_{i+1})$ respectively mean the penalty time for disassembly direction changes, the penalty time for disassembly tool changes and the moving time between part $x_i$ and part $x_{i+1}$.

$$mt = \begin{array}{c} \\ A \\ B \\ C \\ D \\ E \end{array} \begin{array}{c} \begin{array}{ccccc} A & B & C & D & E \end{array} \\ \begin{bmatrix} 0 & 2 & 3 & 4 & 5 \\ 2 & 0 & 1 & 2 & 3 \\ 3 & 1 & 0 & 1 & 4 \\ 4 & 2 & 1 & 0 & 2 \\ 5 & 3 & 4 & 2 & 0 \end{bmatrix} \end{array} \tag{14}$$

$$TT = \begin{array}{c} \\ Ta \\ Tb \\ Tc \\ Td \\ Te \end{array} \begin{array}{c} \begin{array}{ccccc} Ta & Tb & Tc & Td & Te \end{array} \\ \begin{bmatrix} 0 & 1 & 1 & 2 & 2 \\ 1 & 0 & 1 & 2 & 2 \\ 1 & 1 & 0 & 2 & 2 \\ 2 & 2 & 2 & 0 & 1 \\ 2 & 2 & 2 & 1 & 0 \end{bmatrix} \end{array} \tag{15}$$

$$\begin{aligned} fitness &= \sum_{i=0}^{n-2} dt(x_i, x_{i+1}) + \sum_{i=0}^{n-2} tt(x_i, x_{i+1}) + \sum_{i=0}^{n-2} mt(x_i, x_{i+1}) \\ &= (1+1+1+0) + (0+2+0+1) + (1+3+4+2) \\ &= 16s \end{aligned} \tag{16}$$

An example is used to calculate the fitness value of the disassembly sequence. If the disassembly sequence is B/C/A/D/E, the corresponding disassembly direction is X+/Z+/X+/Z+/Z+ (Equation (8) is used for calculate the penalty time for disassembly

direction changes). The corresponding disassembly tool is *Ta*/*Ta*/*Td*/*Td*/*Te* (Equation (15) is used for calculate the penalty time for disassembly tool changes, *Ta*, *Tb*, *Tc*, *Td* and *Te* respectively mean spanner-I, spanner-II, spanner-III, gripper-I and gripper-II). The moving time is described by Equation (14). Because the basic disassembly time is assumed to be constant, only latter three factors in Equation (13) are variable factors, only the latter three factors in Equation (13) are calculated. The fitness value of this example is calculated by Equation (16).

## 5. Robotic disassembly sequence planning using EDBA

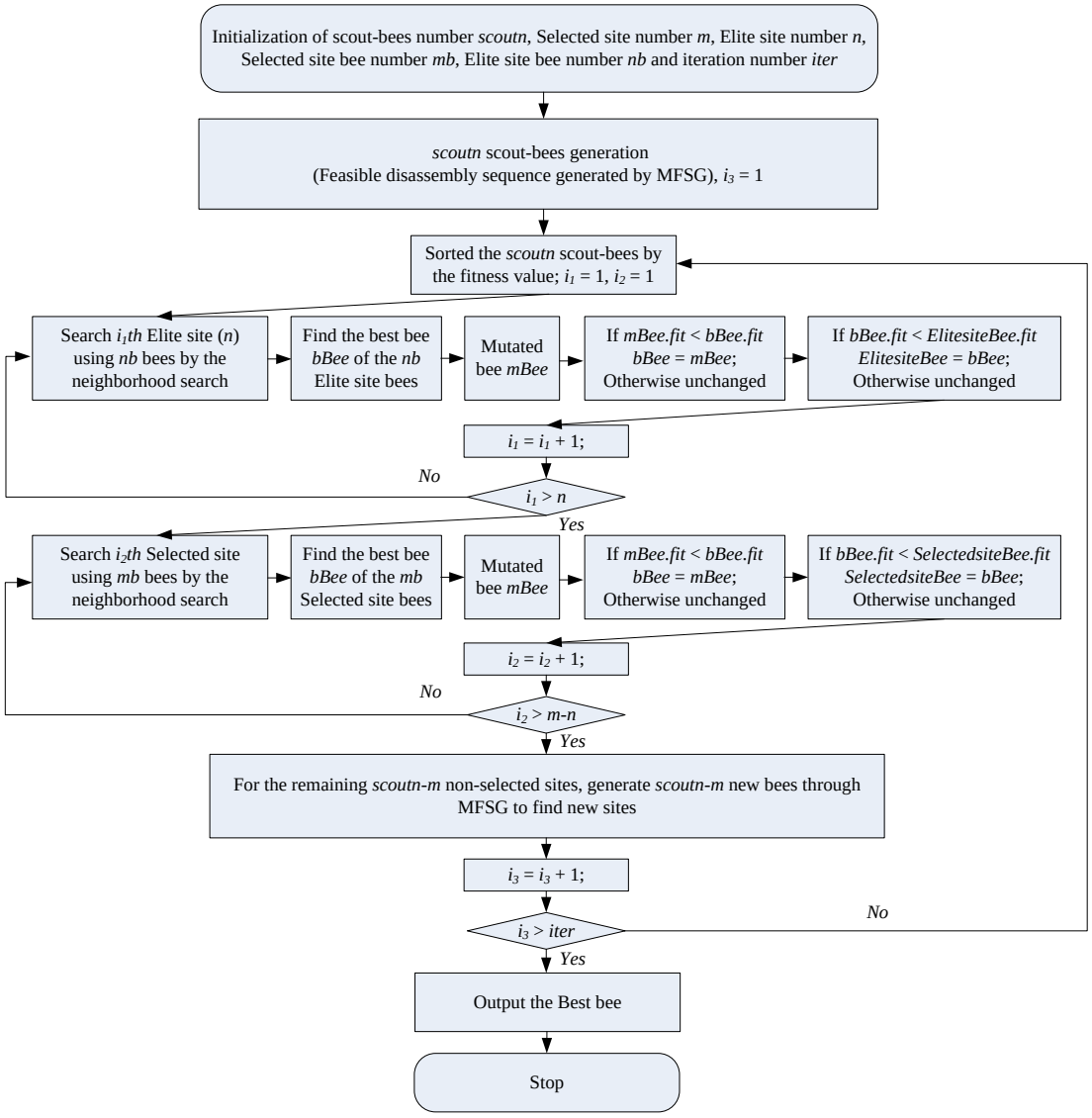


Figure 3. The flow chart of EDBA

The BA is inspired by the foraging process of honey bees. In this paper, the swap/insert operators are used to be the neighborhood search strategies. Similar with the other optimization algorithms, it is easy to get stuck in the local optimum for BA. Aiming at this, the mutation operator is integrated to improve the quality of solutions. The neighborhood search strategy mentioned in Section 5.2 is used to obtain new disassembly solutions and mutation operator proposed here is used to further improve the quality of solutions. The flow chart of EDBA is described in Figure 3.

Firstly, the scout-bees number *scoutn*, selected site number *m*, elite site number *n*, selected site bee number *mb*, elite site bee number *nb* and interation number *iter* are initialized. After that, MFSG is used to generate the feasible disassembly sequences to ensure that all the bees are feasible solutions. Under this condition, *scoutn* scout bees which indicate the feasible disassembly sequences are generated by MFSG. These bees are sent to find the nectar sources (sites) and sorted by fitness value. The nectar sources found by the best *n* scout bees and *m* scout bees are selected as the elite sites and the selected sites respectively. For each elite site, *nb* elite site bees are obtained by neighborhood strategy. Then, mutation operator acts on the best bee of *nb* elite site bees to obtain the mutated bee. If the performance of mutated bee is better than the best Bee, the best bee is replaced the mutated bee, otherwise, it remains unchanged. After that, if the performance of best bee is better than the elite site, the elite site is replaced by the best bee, otherwise, it remains unchanged. For *m*-*n* selected sites (non-elite sites but the selected sites), the process is similar with the elite sites. For the remaining *scoutn*-*m*

non-selected sites, in order to avoid trapping into local optimal solutions, MFSG is used to generate *scoutn-m* new bees which are all feasible solutions to find new sites.

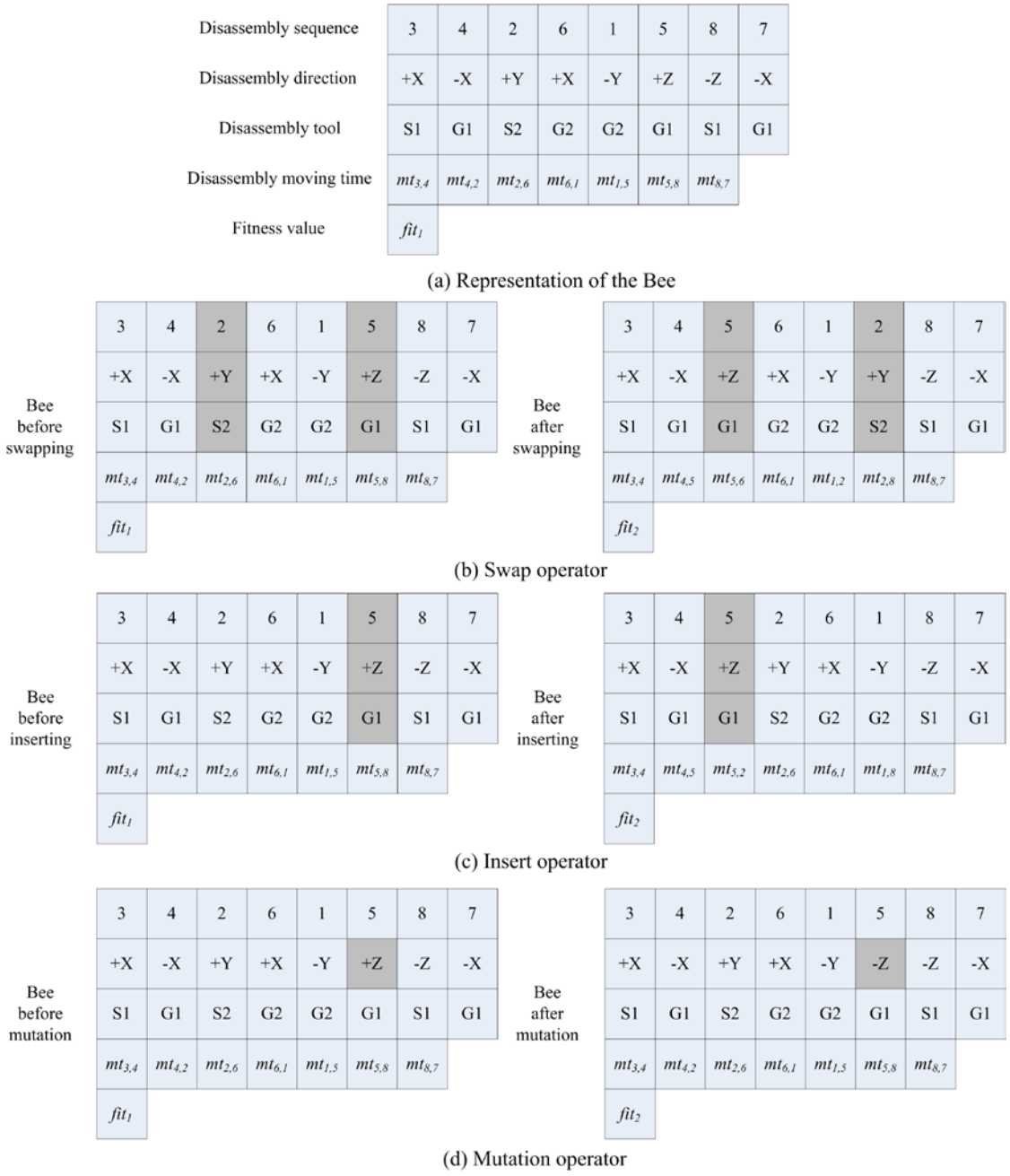



Figure 4. Representation of the Bee, the swap, insert and mutation operators

### *5.1 Initialization of Bees*

A bee which indicates a feasible disassembly solution is represented in Figure 4(a). In this paper, the disassembly sequence generation algorithm is only used for the

purpose of generating feasible disassembly solutions. The feasible disassembly sequence and corresponding disassembly direction are obtained by MFSG. The disassembly moving time array is determined by the predefined length matrix *MP* (calculated by the Equation (11)), linear velocity of industrial robot's end-effector and the disassembly sequence. The fitness value is calculated by Equation (13).

### *5.2 Neighborhood search strategy*

The neighborhood search strategies which contain swap and insert operators are used as shown in Figure 4(b) and 4(c).

The swap operator randomly generates two integers which indicate swap locations of the Bee. A new Bee is obtained by exchanging the elements of selected bits as shown in Figure 4(b). The insert operator randomly chooses one bit from the Bee and inserts the chosen bit to a random location of the Bee as shown in Figure 4(c). The swap/insert operators generate new solutions, but the new solutions may not meet disassembly precedence relationships. Hence, after the new Bee is obtained, its feasibility should be checked by MFSG. If the new Bee is an infeasible solution, the neighborhood search strategy should act on the Bee again until the new Bee is a feasible solution.

### *5.3 Mutation operator*

During the disassembly process, disassembling a component may have several disassembly directions. Based on the analysis and models in Section 3.2, after bolt B, bolt C, component A and component D have been removed, several disassembly directions can be used for disassembling component E. If the disassembly sequence is

B/C/A/D/E, the corresponding disassembly direction can be X+/Z+/X+/Z+/Z+ or X+/Z+/X+/Z+/Z- *etc*. Different disassembly directions for disassembling component E make the industrial robots take different time to adjust its posture. Thus, the mutation operator is added here to increase the diversity of solutions. As shown in Figure 4(d), the mutation operator acts on a random bit of disassembly direction array and then the corresponding bit changes 180 degrees (*eg.* from -Z to +Z). After that, the feasibility of new Bee should be checked as the same way in Section 5.2.

### *5.4 Global search strategy*

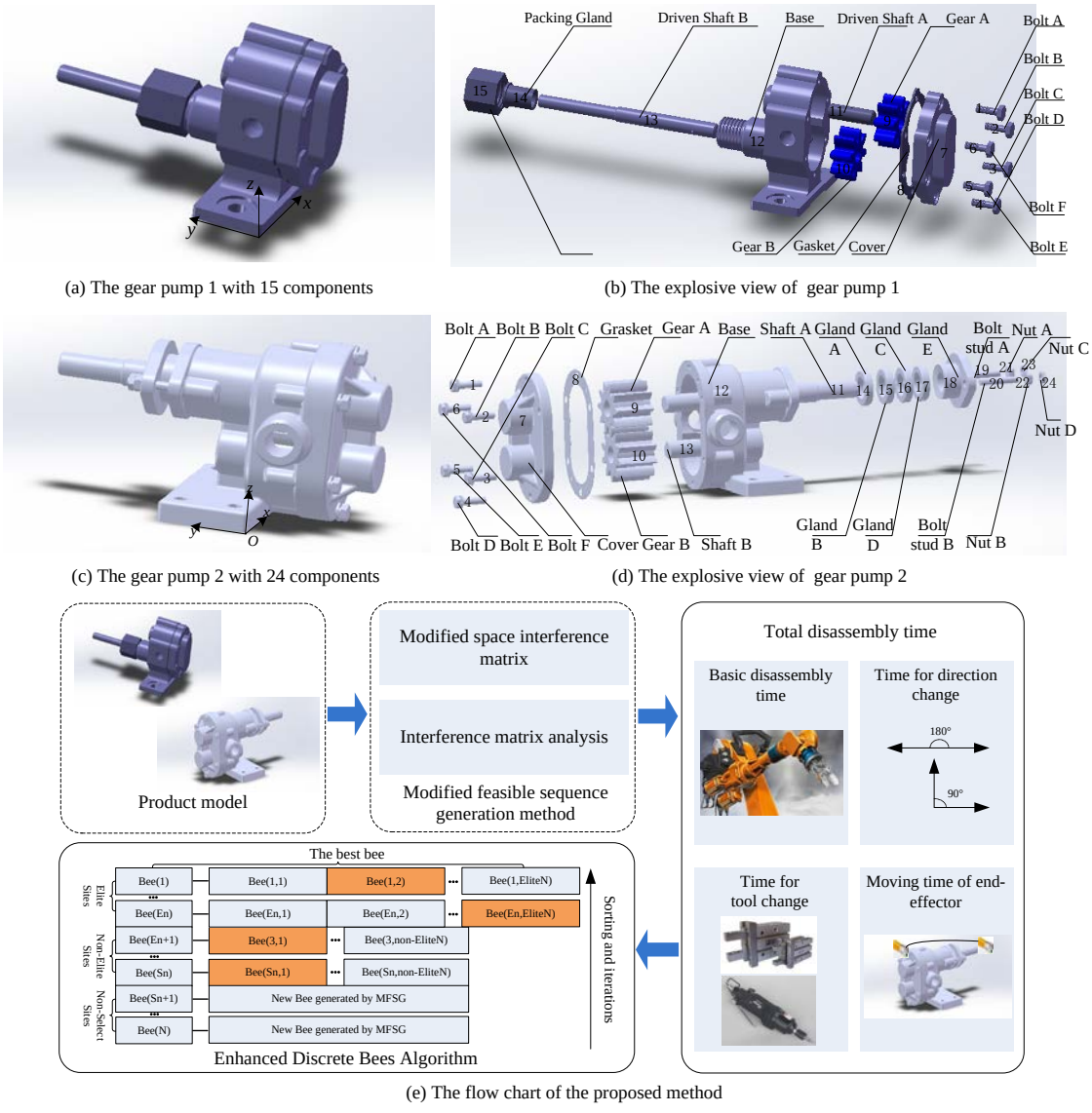
The global search strategy is used to avoid trapping into local optimal solutions. For the remaining *scoutn-n* non-selected sites, *scoutn-n* new Bees are obtained by MFSG and they are dispatched to find new sites.

## 6. Case study and performance analysis

### *6.1 Case study*

In this paper, two gear pumps are used to verify the proposed method as shown in Figure 5(a) and 5(c) and the exploded drawings are shown in Figure 5(b) and 5(d). The properties of all the components are listed in Table 1. The flow chart of the proposed method is shown in Figure 5(e).

The basic disassembly time of each component is assumed to be constant. The penalty time for disassembly direction change is calculated by Equation (8). Three types




(a) The gear pump 1 with 15 components

(b) The explosive view of gear pump 1

(c) The gear pump 2 with 24 components

(d) The explosive view of gear pump 2

(e) The flow chart of the proposed method

Figure 5. The two gear pumps and flow chart of the proposed method

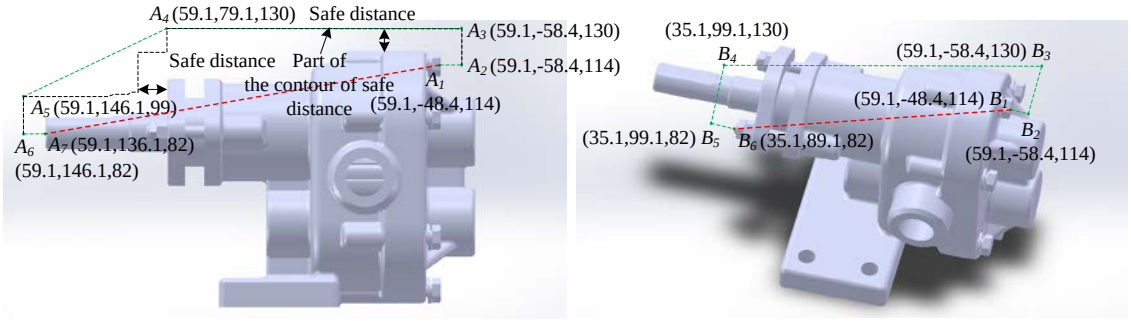
of spanners and two types of grippers are used due to different sizes and different types of components. The penalty time for disassembly tool changes is described by Equation (15). The safe distance is set to 10 *mm* and the length matrix $MP_{gp}$ is represented by Equation (17). In this matrix, element $a_{m,n}$ means the length of moving path between disassembly points of component *m* and component *n*. Based on the gear pump 2 (Hoge 2017), from Figure 6(a), it is obvious that $a_{1,11}$ (manually calculated by Equation (18)) is made up of several coplanar line segments. In addition, from Figure 6(b), $a_{1,19}$ is

calculated by the summation of length of several non-coplanar line segments as shown in Equation (19). According to Equation (20), the moving time $mt_{1,11}$ is calculated by $a_{1,11}$ and the linear velocity of industrial robot's end-effector $v_{end\text{-}effector}$ which is assumed to be 12 *mm*/s.

Table 1. The properties of all components of the two gear pumps

| Gear pump | Number | Disassembly task | Basic disassembly time | Disassembly tool | Disassembly point (mm) |
|---|---|---|---|---|---|
| 1 | 1 | Unscrew the **Bolt A** | $bt_{1\text{-}1}$ | Spanner-I (*Ta*) | [49.4, -12.6, 105.5] |
| | 2 | Unscrew the **Bolt B** | $bt_{1\text{-}2}$ | Spanner-I (*Ta*) | [74.4, -12.6, 81] |
| | 3 | Unscrew the **Bolt C** | $bt_{1\text{-}3}$ | Spanner-I (*Ta*) | [74.4, -12.6, 45] |
| | 4 | Unscrew the **Bolt D** | $bt_{1\text{-}4}$ | Spanner-I (*Ta*) | [49.4, -12.6, 20.5] |
| | 5 | Unscrew the **Bolt E** | $bt_{1\text{-}5}$ | Spanner-I (*Ta*) | [24.4, -12.6, 45] |
| | 6 | Unscrew the **Bolt F** | $bt_{1\text{-}6}$ | Spanner-I (*Ta*) | [24.4, -12.6, 81] |
| | 7 | Remove the **Cover** | $bt_{1\text{-}7}$ | Gripper-II (*Te*) | [49.4, -20.6, 63] |
| | 8 | Remove the **Gasket** | $bt_{1\text{-}8}$ | Gripper-I (*Td*) | [49.4, 1.4, 105.5] |
| | 9 | Remove the **Gear A** | $bt_{1\text{-}9}$ | Gripper-I (*Td*) | [49.4, 3.4, 81] |
| | 10 | Remove the **Gear B** | $bt_{1\text{-}10}$ | Gripper-I (*Td*) | [49.4, 3.4, 45] |
| | 11 | Remove the **Driven Shaft A** | $bt_{1\text{-}11}$ | Gripper-I (*Td*) | [49.4, -7.6, 81] |
| | 12 | Remove the **Base** | $bt_{1\text{-}12}$ | Gripper-II (*Te*) | [49.4, 49.4, 81] |
| | 13 | Remove the **Driven Shaft B** | $bt_{1\text{-}13}$ | Gripper-I (*Td*) | [49.4, 152.4, 45] |
| | 14 | Remove the **Packing Gland** | $bt_{1\text{-}14}$ | Gripper-I (*Td*) | [49.4, 91.4, 45] |
| | 15 | Unscrew the **Gland Nut** | $bt_{1\text{-}15}$ | Spanner-II (*Tb*) | [49.4, 96.4, 45] |
| 2 | 1 | Unscrew the **Bolt A** | $bt_{2\text{-}1}$ | Spanner-I (*Ta*) | [59.1,-48.4,114] |
| | 2 | Unscrew the **Bolt B** | $bt_{2\text{-}2}$ | Spanner-I (*Ta*) | [90.3,-48.4,89] |
| | 3 | Unscrew the **Bolt C** | $bt_{2\text{-}3}$ | Spanner-I (*Ta*) | [90.3,-48.4,33] |
| | 4 | Unscrew the **Bolt D** | $bt_{2\text{-}4}$ | Spanner-I (*Ta*) | [59.1,-48.4,8] |
| | 5 | Unscrew the **Bolt E** | $bt_{2\text{-}5}$ | Spanner-I (*Ta*) | [27.9,-48.4,33] |
| | 6 | Unscrew the **Bolt F** | $bt_{2\text{-}6}$ | Spanner-I (*Ta*) | [27.9,-48.4,89] |
| | 7 | Remove the **Cover** | $bt_{2\text{-}7}$ | Gripper-II (*Te*) | [59.1,-64.4,82] |
| | 8 | Remove the **Gasket** | $bt_{2\text{-}8}$ | Gripper-I (*Td*) | [59.1,-31.4,114] |
| | 9 | Remove the **Gear A** | $bt_{2\text{-}9}$ | Gripper-I (*Td*) | [59.1,-30.9,82] |
| | 10 | Remove the **Gear B** | $bt_{2\text{-}10}$ | Gripper-I (*Td*) | [59.1,-30.9,40] |
| | 11 | Remove the **Shaft A** | $bt_{2\text{-}11}$ | Gripper-I (*Td*) | [59.1,136.1,82] |
| | 12 | Remove the **Base** | $bt_{2\text{-}12}$ | Gripper-II (*Te*) | [59.1,7.1,114] |
| | 13 | Remove the **Shaft B** | $bt_{2\text{-}13}$ | Gripper-I (*Td*) | [59.1,-48.9,40] |
| | 14 | Remove the **Gland A** | $bt_{2\text{-}14}$ | Gripper-I (*Td*) | [59.1,34.1, 94.8] |
| | 15 | Remove the **Gland B** | $bt_{2\text{-}15}$ | Gripper-I (*Td*) | [59.1,41.1, 94.8] |
| | 16 | Remove the **Gland C** | $bt_{2\text{-}16}$ | Gripper-I (*Td*) | [59.1,48.1, 94.8] |

| | | | | | |
|---|---|---|---|---|---|
| | 17 | Remove the **Gland D** | $bt_{2\text{-}17}$ | Gripper-I (*Td*) | [59.1,55.1,94.8] |
| | 18 | Remove the **Gland E** | $bt_{2\text{-}18}$ | Gripper-I (*Td*) | [59.1,79.1,82] |
| | 19 | Unscrew the **Bolt stud A** | $bt_{2\text{-}19}$ | Spanner-II (*Tb*) | [35.1,89.1,82] |
| | 20 | Unscrew the **Bolt stud B** | $bt_{2\text{-}20}$ | Spanner-II (*Tb*) | [83.1,89.1,82] |
| | 21 | Unscrew the **Nut A** | $bt_{2\text{-}21}$ | Spanner-III (*Tc*) | [35.1,84.1,82] |
| | 22 | Unscrew the **Nut B** | $bt_{2\text{-}22}$ | Spanner-III (*Tc*) | [83.1,84.1,82] |
| | 23 | Unscrew the **Nut C** | $bt_{2\text{-}23}$ | Spanner-III (*Tc*) | [35.1,87.1,82] |
| | 24 | Unscrew the **Nut D** | $bt_{2\text{-}24}$ | Spanner-III (*Tc*) | [83,1,87.1,82] |




(a) The moving path between bolt A and shaft A

(b) The moving path between bolt A and bolt stud A

Figure 6. The moving path between different disassembly points

$$MP_{gp1} = \left[ a_{i,j} \right], MP_{gp2} = \left[ a_{m,n} \right] \; i, j \in [1,15], m, n \in [1,24] \tag{17}$$

$$a_{1,11} = A_1A_2 + A_2A_3 + A_3A_4 + A_4A_5 + A_5A_6 + A_6A_7 = 10 + 16 + 137.5 + 73.8 + 17 + 10 = 264.3mm \tag{18}$$

$$a_{1,19} = B_1B_2 + B_2B_3 + B_3B_4 + B_4B_5 + B_5B_6 = 10 + 16 + 159.3 + 48 + 10 = 243.3mm \tag{19}$$

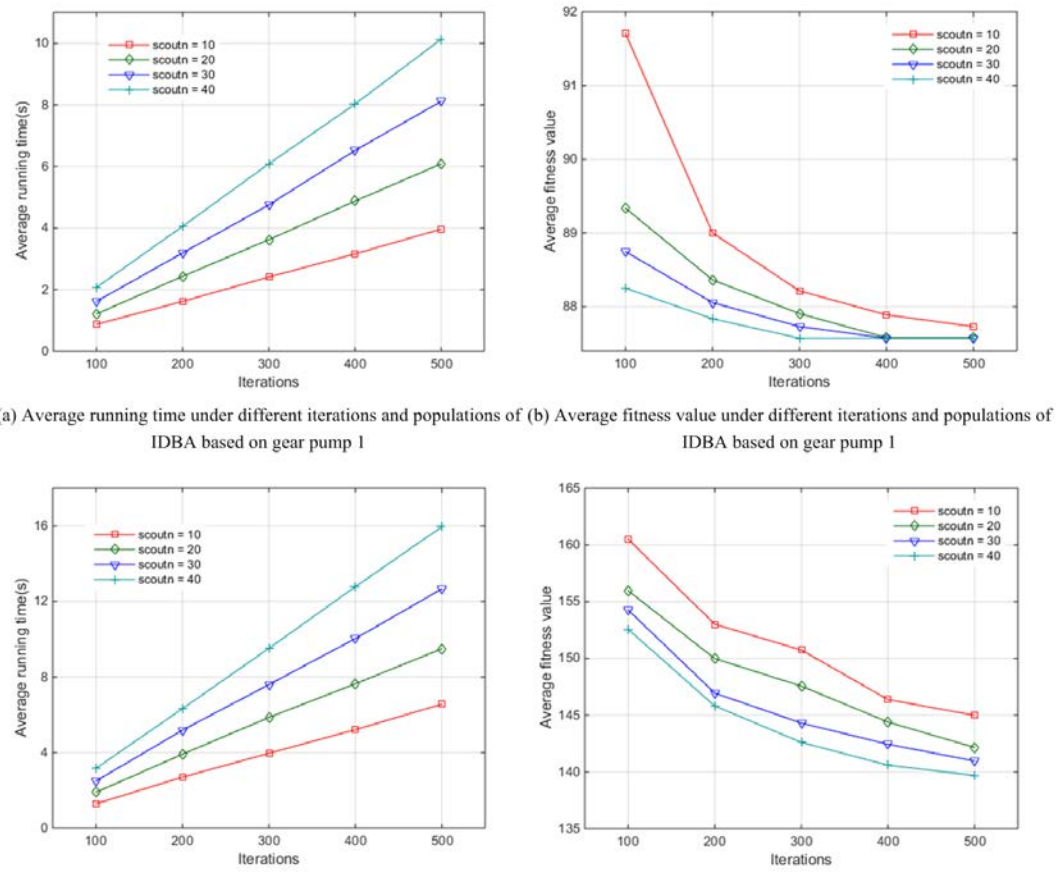
$$mt_{1,11} = a_{1,11} / v_{end-effector} = 264.3mm / 12mm / s = 22.025s \tag{20}$$

### *6.2 Performance analysis*

In this section, simulations were taken on PC with 2.30GHz Intel core i5-6200U CPU, 4 GB memory based on Matlab 2014b. This section contains three parts: 1. the performance analysis under different iterations and populations of EDBA; 2. the comparisons of results obtained by different methods; 3. the performance comparative analysis between EDBA and the other optimization algorithms.

For the performance analysis under different iterations and populations of EDBA, selected site number *m*, elite site number *n*, selected site bee number *mb*, elite site bee

number *nb* were set to 4, 1, 1 and 2 respectively, the running time and the fitness value are analyzed under different iterations (100, 200, 300, 400 and 500) and different scout bees (10, 20, 30 and 40). Each simulation was repeated 10 times. From Figure 7(a) and 7(c), when the iteration number *iter* is fixed, the running time of EDBA increases linearly as the number of scout bees, when the number of scout bees *scoutn* is fixed, the running time of EDBA linearly increases with iteration number. The average fitness values under different iterations are shown in Figure 7(b) and 7(d). When *scoutn* is 10 and *iter* is 100, it has the worst performance because there are insufficient scout bees and iterations for EDBA to obtain high quality solutions. It is obvious that the quality of solutions obtained by EDBA increases with iterations and populations.




(a) Average running time under different iterations and populations of IDBA based on gear pump 1

(b) Average fitness value under different iterations and populations of IDBA based on gear pump 1

(c) Average running time under different iterations and populations of IDBA based on gear pump 2

(d) Average fitness value under different iterations and populations of IDBA based on gear pump 2

Figure 7. The performance of EDBA under different iterations and populations

Alshibli et al. (2015) used the Euclidean distance (the red dotted line in Figure (6) to calculate the moving time between different disassembly points. This method is also applied on the gear pumps to obtain the optimal disassembly sequence, the result (Result 1) is compared with the optimal solution (Result 2) obtained by proposed method in this paper. Due to the large solution space, it is difficult to exhaust all the solutions to get the best solutions. Thus, the near-optimal solutions are obtained by the following methods. Iteration number *iter*, scout bees number *scoutn*, selected site number *m*, elite site number *n*, selected site bee number *mb*, elite site bee number *nb* of EDBA were set to 500, 40, 4, 1, 1 and 2 respectively. Simulations were repeated 1000 times. The solution with the minimum fitness value is the near-optimal solution. The results are shown in Table 2 (‘1’ and ‘2’ mean ‘Y+’ and ‘Y-’ respectively). From Table 2, it is concluded that Result 1 is obviously different from Result 2 for both the two gear pumps. Although the fitness value of Result 1 is smaller than Result 2, Result 1 is obtained by using the Euclidean distance to calculate the moving time. It is impractical for the industrial robot’s end-effector to move straight-line between different disassembly points, it ignores the obstacle caused by contour of EoL product. Thus, Result 2 obtained by proposed method is more suitable for practical robotic disassembly.

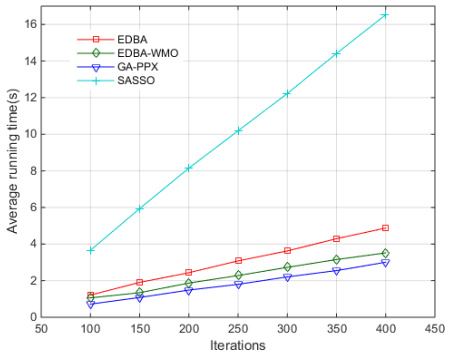
Table 2. Comparison of results obtained by different methods

| Gear pump | Result | | |
|---|---|---|---|
| 1 | 1 | Disassembly sequence | 5-4-3-2-1-6-7-11-8-9-10-13-15-14-12 |
| | | Disassembly direction | 2-2-2-2-2-2-2-2-2-2-2-2-1-1-1 |
| | | Fitness value | 58.5038 |

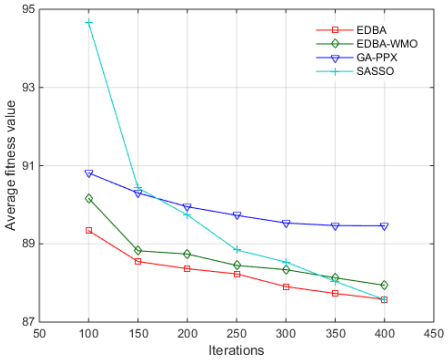
| | | | |
|---|---|---|---|
| Gear pump 2 | Result 2 | Disassembly sequence | 5-4-3-2-1-6-7-10-9-11-8-13-15-14-12 |
| | | Disassembly direction | 2-2-2-2-2-2-2-2-2-2-2-2-1-1-1 |
| | | Fitness value | 87.5731 |
| | Result 1 | Disassembly sequence | 24-22-20-23-21-19-18-11-17-16-15-14-1-6-5-4-3-2-7-13-10-9-8-12 |
| | | Disassembly direction | 1-1-1-1-1-1-1-1-1-1-1-1-2-2-2-2-2-2-2-2-2-2-2-2 |
| | | Fitness value | 77.8101 |
| | Result 2 | Disassembly sequence | 24-22-20-23-21-19-1-2-3-4-5-6-7-13-10-9-8-12-14-15-16-17-18-11 |
| | | Disassembly direction | 1-1-1-1-1-1-2-2-2-2-2-2-2-2-2-2-2-2-2-2-2-1-1 |
| | | Fitness value | 135.3167 |

In addition, the performance of EDBA, EDBA without mutation operator (EDBA-WMO), Genetic Algorithm with Precedence Preserve Crossover (GA-PPX) (Kheder, Trigui, and Aifaoui 2015) and Self-Adaptive Simplified Swarm Optimization (SASSO) (Yeh 2012) are compared. Based on the two gear pumps, for all the simulations, selected site number *m*, elite site number *n*, selected site bee number *mb*, elite site bee number *nb* of EDBA and EDBA-WMO were set to 4, 1, 1 and 2 respectively. The mutation ratio of GA-PPX was set to 0.1 respectively. The parameters $C_g$, $C_p$ and $C_w$ of SASSO were controlled by self-adaptive parameter control (Yeh 2012). Each simulation was repeated 10 times. For the gear pump 1, the populations of all the optimization algorithms were set to 20, results under different iterations (from 100 to 400) are compared as shown in Figure 8(a) and 8(b). For the gear pump 2, the populations of all the optimization algorithms were set to 40, results under different iterations (from 100 to 600) are compared as shown in Figure 8(e) and 8(f). From

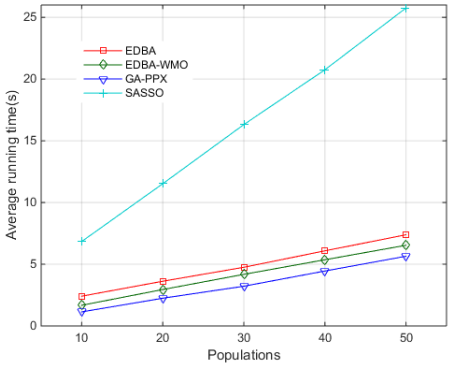
Figure 8(a) and 8(e), GA-PPX and SASSO need the least running time and the most running time among all the algorithms respectively. EDBA needs more running time than EDBA-WMO, because EDBA needs more time on mutation operator. From Figure 8(b) and 8(f), it is obvious that the solutions obtained by EDBA have the smallest average fitness value than the others. In addition, the quality of solutions obtained by all the optimization algorithms increases with iterations, the quality of solution obtained by SASSO is improved more obviously than GA-PPX. For both the two gear pumps, with the help of mutation operator, EDBA needs more running time but generates better quality of solutions than EDBA-WMO under different iterations. After that, for the gear pump 1, the iteration was set to 300 and simulations under different populations (from 10 to 50) are compared as shown in Figure 8(c) and 8(d). For the gear pump 2, the iteration was set to 500 and simulations under different populations (from 10 to 50) are compared as shown in Figure 8(g) and 8(h). It is obvious that GA-PPX needs the least running time but generates the worst quality of solutions. SASSO needs the most running time while EDBA generates the best quality of solutions. In all situations based on the two gear pumps, EDBA can find the best quality of solutions than the others. From the perspective of iteration process, the iteration and population were set to 300 and 20 respectively for the gear pump 1. For the gear pump 2, the iteration and population were set to 500 and 40 respectively. Simulations were repeated 10 times, the average fitness value of iterative process is shown in Figure 8(i) and 8(j). It is obvious that SASSO has the slowest convergence speed and EDBA converges to the smallest fitness value than the others.



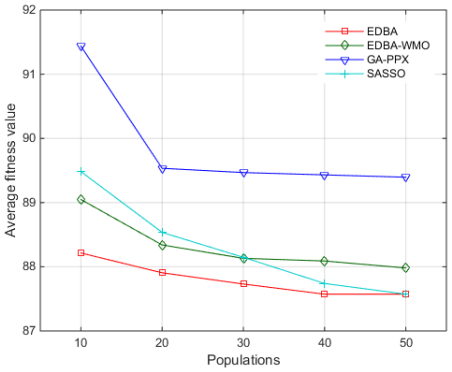

(a) Average running time under different iterations of the four optimization algorithms based on gear pump 1



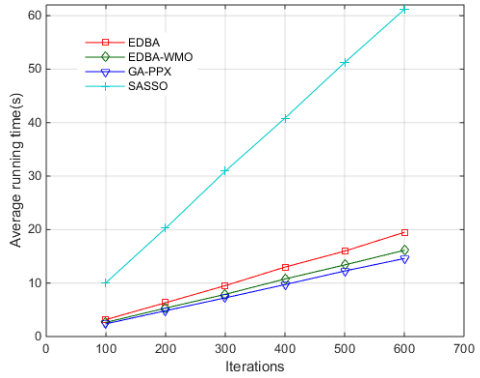

(b) Average fitness value under different iterations of the four optimization algorithms based on gear pump 1



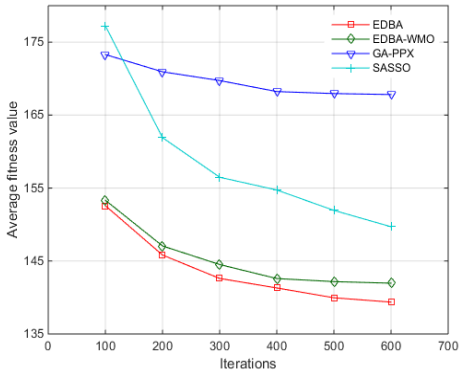

(c) Average running time under different populations of the four optimization algorithms based on gear pump 1



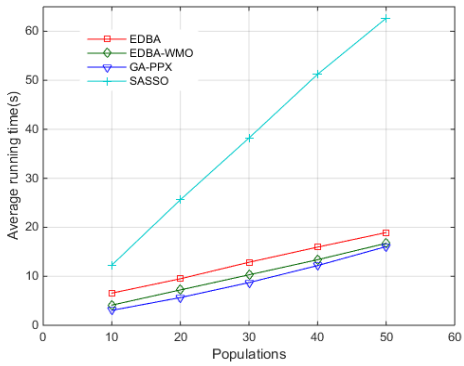

(d) Average fitness value under different populations of the four optimization algorithms based on gear pump 1



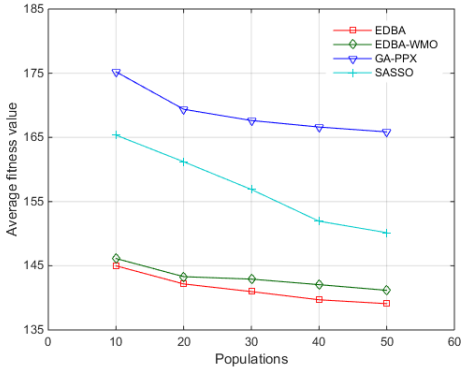

(e) Average running time under different iterations of the four optimization algorithms based on gear pump 2



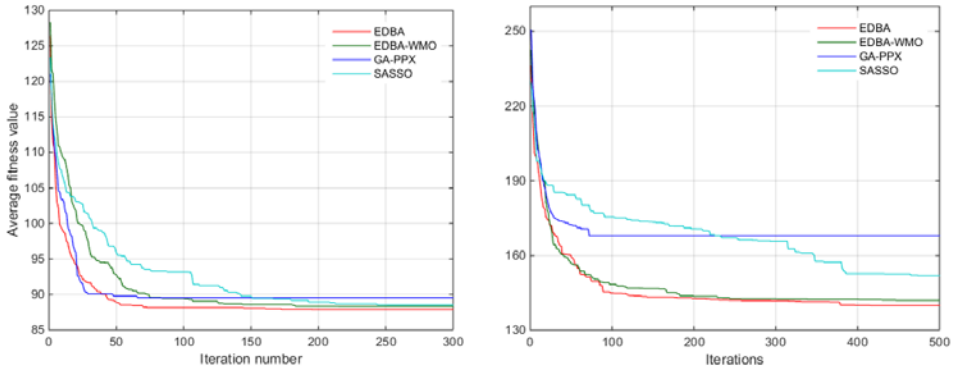

(f) Average fitness value under different iterations of the four optimization algorithms based on gear pump 2




(g) Average running time under different populations of the four optimization algorithms based on gear pump 2




(h) Average fitness value under different populations of the four optimization algorithms based on gear pump 2




(i) Iteration process of the four optimization algorithms based on gear pump 1

(j) Iteration process of the four optimization algorithms based on gear pump 2

Figure 8. The performance comparisons of four optimization algorithms

## 7. Conclusion

In this paper, RDSP was solved by EDBA. Firstly, MFSG was used to describe disassembly precedence relationships of EoL product. After that, the evaluation criterions for RDSP were proposed. Rather than Euclidean distance, considering the safe distance between industrial robot's end-effector and contour of EoL product, the obstacle-avoidance moving path was used to calculate the moving time between different disassembly points. Afterwards, based on MFSG, to minimize the total disassembly time, EDBA was proposed to solve RDSP. Simulations were carried out to verify the proposed method. The result shows the proposed method is more adaptable for RDSP compared with the traditional method. With the help of mutation operator, EDBA can obtain better quality of solutions than the others. However, in this paper, it is time-consuming to manually calculate the length matrix $MP_{gp}$ especially when the number of component increases. In the future, we will study on how to efficiently obtain the length matrix $MP_{gp}$. In addition, the moving time between different disassembly points depends on the factors such as robot types, the position of EoL products *etc.*, the future work will include kinematics parameters of industrial robots with RDSP to find

the optimal disassembly sequence. Besides, because the proposed method is only feasible under certain assumptions mentioned in Section 3, in the future, we will also add the machine vision systems in robotic disassembly systems to handle uncertain problems in disassembly process.

## Disclosure statement

No potential conflict of interest was reported by the authors.


## Acknowledgements

This work is supported by National Natural Science Foundation of China (Grant Nos. 51775399 and 51475343), the Keygrant Project of Hubei Technological Innovation Special Fund (Grant No. 2016AAA016), Engineering and Physical Sciences Research Council (EPSRC), UK (Grant No. EP/N018524/1), and the China Scholarship Council (CSC).


## References


Alshibli, M., A. ElSayed, E. Kongar, T. M. Sobh, and S. M. Gupta. 2015. “Disassembly sequencing using tabu search.” *Journal of Intelligent & Robotic Systems* 82 (1): 1-11. doi: 10.1007/s10846-015-0289-9.

Constantinescu, D., and E. A. Croft. 2000. “Smooth and time-optimal trajectory planning for industrial manipulators along specified paths.” *Journal of robotic systems* 17 (5): 233-249. doi: 10.1002/(SICI)1097-4563(200005)17:5<233::AID-ROB1>3.0.CO;2-Y.

Diallo, C., U. Venkatadri, A. Khatab, and S. Bhakthavatchalam. 2017. “State of the art

review of quality, reliability and maintenance issues in closed-loop supply chains with remanufacturing." *International Journal of Production Research*, 55 (5): 1277-1296. doi: 10.1080/00207543.2016.1200152

ElSayed, A., E. Kongar, and S. M. Gupta. 2010. "A Genetic Algorithm Approach To End-Of-Life Disassembly Sequencing for Robotic Disassembly." In *Proceedings of the 2010 Northeast Decision Sciences Institute Conference*. 402-408. Alexandria.

ElSayed, A., E. Kongar, S. M. Gupta, and T. Sobh. 2011. "An online genetic algorithm for automated disassembly sequence generation." In *Proceedings of ASME 2011 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, 657-664. Washington: American Society of Mechanical Engineers. doi: 10.1115/DETC2011-48635.

ElSayed, A., E. Kongar, S. M. Gupta, and T. Sobh. 2012. "A robotic-driven disassembly sequence generator for end-of-life electronic products." *Journal of Intelligent & Robotic Systems* 68 (1): 43-52. doi: 10.1007/s10846-012-9667-8.

Go, T. F., D. A. Wahab, M. A. Rahman, R. Ramli, and A. Hussain. 2012. "Genetically optimised disassembly sequence for automotive component reuse." *Expert Systems with Applications* 39 (5): 5409-5417. doi: 10.1016/j.eswa.2011.11.044.

Guide, V. D. R. 2000. "Production planning and control for remanufacturing: industry practice and research needs." *Journal of Operations Management* 18 (4): 467-483. doi: 10.1016/S0272-6963(00)00034-6.

Hoge. 2017. "Gear pump (10L/min)." GRABCAD COMMUNITY. Accessed February 3 2017. https://grabcad.com/library/gear-pump-10l-min-1

Jin, G. Q., W. D. Li, and K. Xia. 2013. “Disassembly matrix for liquid crystal displays televisions.” *Procedia CIRP* 11: 357-362. doi: 10.1016/j.procir.2013.07.015.

Jin, G. Q., W. D. Li, S. Wang, and S. M. Gao. 2015. “A systematic selective disassembly approach for Waste Electrical and Electronic Equipment with case study on liquid crystal display televisions.” *Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture* Special issue: 1-18. doi: 10.1177/0954405415575476.

Kheder, M., M. Trigui, and N. Aifaoui. 2015. “Disassembly sequence planning based on a genetic algorithm.” *Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science* 229 (12): 2281-2290. doi: 10.1177/0954406214557340.

Kim, H. W., and D. H. Lee. 2017. “An optimal algorithm for selective disassembly sequencing with sequence-dependent set-ups in parallel disassembly environment.” *International Journal of Production Research*, 1-17. doi: 10.1080/00207543.2017.1342879

Lambert, A. J. D. 2003. “Disassembly sequencing: a survey.” *International Journal of Production Research* 41 (16): 3721-3759. doi: 10.1080/0020754031000120078.

Luo, Y. T., Q. J. Peng, and P. H. Gu. 2016. “Integrated multi-layer representation and ant colony search for product selective disassembly planning.” *Computers in Industry* 75: 13-26. doi: 10.1016/j.compind.2015.10.011.

Meng, K., P. H. Lou, X. H. Peng, and V. Prybutok. 2016. “An improved co-evolutionary algorithm for green manufacturing by integration of recovery option selection and

disassembly planning for end-of-life products." *International Journal of Production Research*, 54 (18): 5567-5593. doi: 10.1080/00207543.2016.1176263

Percoco, G., and M. Diella. 2013. "Preliminary evaluation of artificial bee colony algorithm when applied to multi objective partial disassembly planning." *Research Journal of Applied Sciences, Engineering and Technology* 6 (17): 3234-3243.

Pham, D. T., and A. Ghanbarzadeh. 2007. "Multi-objective optimisation using the bees algorithm." In *Proceedings of IPROMS 2007 Conference*. 529-533. Cardiff.

Ren, Y. P., D. Y. Yu, C. Y. Zhang, G. D. Tian, L. L. Meng, and X. Q. Zhou. 2017. "An improved gravitational search algorithm for profit-oriented partial disassembly line balancing problem." *International Journal of Production Research*, 1-15. doi: 10.1080/00207543.2017.1341066

Song, X. W., W. D. Zhou, X. X. Pan, and K. Feng. 2014. "Disassembly sequence planning for electro-mechanical products under a partial destructive mode." *Assembly Automation* 34 (1): 106-114. doi: 10.1108/AA-01-2013-006.

Tao, F., D. M. Zhao, Y. F. Hu, and Z. D. Zhou. 2008. "Resource service composition and its optimal-selection based on particle swarm optimization in manufacturing grid system." *IEEE Transactions on industrial informatics* 4 (4): 315-327. doi: 10.1109/TII.2008.2009533.

Tao, F., J. F. Cheng, Q. L. Qi, M. Zhang, H. Zhang, and F. Y. Sui. 2017a. "Digital twin-driven product design, manufacturing and service with big data." *The International Journal of Advanced Manufacturing Technology* 1-14. doi: 10.1007/s00170-017-0233-1.

Tao, F., J. F. Cheng, Y. Cheng, S. X. Gu, T. Y. Zheng, and H. Yang. 2017b. “SDMSim: a manufacturing service supply–demand matching simulator under cloud environment.” *Robotics and Computer-Integrated Manufacturing* 45: 34-46. doi: 10.1016/j.rcim.2016.07.001.

Tian, G. D., M. C. Zhou, and J. W. Chu. 2013. “A chance constrained programming approach to determine the optimal disassembly sequence.” *IEEE Transactions on Automation Science and Engineering* 10 (4): 1004-1013. doi: 10.1109/TASE.2013.2249663.

Vongbunyong, S., S. Kara, and M. Pagnucco. 2012. “A framework for using cognitive robotics in disassembly automation.” In *Proceedings of the 19th CIRP International Conference on Life Cycle Engineering*, 173-178. Berkeley: Springer. doi: 10.1007/978-3-642-29069-5_30.

Vongbunyong, S., S. Kara, and M. Pagnucco. 2013a. “Basic behaviour control of the vision-based cognitive robotic disassembly automation.” *Assembly Automation* 33 (1): 38-56. doi: 10.1108/01445151311294694.

Vongbunyong, S., S. Kara, and M. Pagnucco. 2013b. “Application of cognitive robotics in disassembly of products.” *CIRP Annals-Manufacturing Technology* 62 (1): 31-34. doi: 10.1016/j.cirp.2013.03.037.

Vongbunyong, S., S. Kara, and M. Pagnucco. 2015. “Learning and revision in cognitive robotics disassembly automation.” *Robotics and Computer-Integrated Manufacturing* 34: 79-94. doi: 10.1016/j.rcim.2014.11.003.

Xia, K., L. Gao, L. H. Wang, W. D. Li, and K. M. Chao. 2013. “A Simplified

Teaching-Learning-Based Optimization Algorithm for Disassembly Sequence Planning." In *Proceedings of 2013 IEEE 10th International Conference on e-Business Engineering*, 393-398. Coventry. doi: 10.1109/ICEBE.2013.60.

Xia, K., L. Gao, W. D. Li, and K. M. Chao. 2014a. "Disassembly sequence planning using a simplified teaching–learning-based optimization algorithm." *Advanced Engineering Informatics* 28 (4): 518-527. doi: 10.1016/j.aei.2014.07.006.

Xia, K., L. Gao, W. D. Li, L. H. Wang, and K. M. Chao. 2014b. "A q-learning based selective disassembly planning service in the cloud based remanufacturing system for weee." In *Proceedings of ASME 2014 International Manufacturing Science and Engineering Conference*, V001T04A012-V001T04A012. Detroit: American Society of Mechanical Engineers. doi: 10.1115/MSEC2014-4008.

Xing, Y. F., C. E. Wang, and Q. Liu. 2012. "Disassembly sequence planning based on pareto ant colony algorithm." *Journal of Mechanical Engineering* 48 (9): 186-192. doi: 10.3901/JME.2012.09.186.

Xu, J., S. Y. Zhang, and S. M. Fei. 2011. "Product remanufacture disassembly planning based on adaptive particle swarm optimization algorithm." *Journal of Zhejiang University. Engineering Science* 45 (10): 1746-1752. doi: 10.3785/j.issn.1008-973X.2011.10.008.

Xu, W. J., S. S. Tian, Q. Liu, Y. Q. Xie, Z. D. Zhou, and D. T. Pham. 2016. "An improved discrete bees algorithm for correlation-aware service aggregation optimization in cloud manufacturing." *The International Journal of Advanced Manufacturing Technology* 84 (1-4): 17-28. doi: 10.1007/s00170-015-7738-2.

Yeh, W. C. 2012. “Optimization of the disassembly sequencing problem on the basis of self-adaptive simplified swarm optimization.” *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans* 42 (1): 250-261. doi: 10.1109/TSMCA.2011.2157135.

Yuce, B., M. S. Packianather, E. Mastrocinque, D. T. Pham, and A. Lambiase. 2013. “Honey bees inspired optimization method: the bees algorithm.” *Insects* 4 (4): 646-662. doi: 10.3390/insects4040646.